



PCT/FR 00 / 0 0 1 5 0

(4)

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

REC'D 14 FEB 2000

WIPO PCT

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 01 FEV. 2000

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

**DOCUMENT DE
PRIORITÉ**
PRÉSENTÉ OU TRANSMIS
CONFORMÉMENT À LA REGLE
17.1.a) OU b)

Martine PLANCHE

INSTITUT
NATIONAL DE
LA PROPRIÉTÉ
INDUSTRIELLE

SIEGE
26 bis, rue de Saint Petersburg
75800 PARIS Cédex 08
Téléphone : 01 53 04 53 04
Télécopie : 01 42 93 59 30

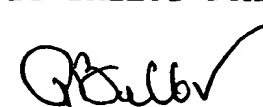
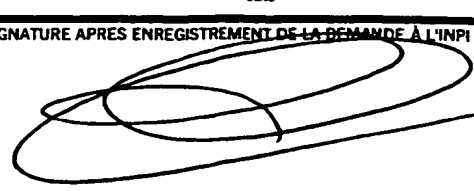
THIS PAGE BLANK (USPTO)

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : (1) 42.94.52.52 Télécopie : (1) 42.93.59.30

REQUÊTE EN DÉLIVRANCE

Confirmation d'un dépôt par télécopie ☐

Cet imprimé est à remplir à l'encre noire en lettres capitales

DATE DE REMISE DES PIÈCES 09/03/99 N° D'ENREGISTREMENT NATIONAL 99 02924 - DÉPARTEMENT DE DÉPÔT 75 DATE DE DÉPÔT 09 MARS 1999		1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE CABINET BALLOT-SCHMIT 7, rue Le Sueur 75116 PARIS FRANCE									
2 DEMANDE Nature du titre de propriété industrielle <input checked="" type="checkbox"/> brevet d'invention <input type="checkbox"/> demande divisionnaire <input type="checkbox"/> certificat d'utilité <input type="checkbox"/> transformation d'une demande de brevet européen <input type="checkbox"/> brevet d'invention <input checked="" type="checkbox"/> demande initiale		n° du pouvoir permanent SM/015031 références du correspondant 01.40.67.11.99 téléphone date									
Établissement du rapport de recherche <input type="checkbox"/> différé <input checked="" type="checkbox"/> immédiat Le demandeur, personne physique, requiert le paiement échelonné de la redevance <input type="checkbox"/> oui <input checked="" type="checkbox"/> non Titre de l'invention (200 caractères maximum) <p style="text-align: center;">Procédé et dispositif de surveillance du déroulement d'un programme, dispositif programmé permettant la surveillance de son programme.</p>											
3 DEMANDEUR (S) n° SIREN : 7.4.9.7.1.1.2.0.0 code APE-NAF : Nom et prénoms (souligner le nom patronymique) ou dénomination GEMPLUS française Nationalité (s)		Forme juridique Société en Commandite par Actions									
Adresse (s) complète (s) Avenue du Pic de Bertagne Parc d'activités de la Plaine de Jouques 13420 GEMPLUS		Pays FRANCE									
4 INVENTEUR (S) Les inventeurs sont les demandeurs <input type="checkbox"/> oui <input checked="" type="checkbox"/> non En cas d'insuffisance de place, poursuivre sur papier libre <input type="checkbox"/> Si la réponse est non, fournir une désignation séparée											
5 RÉDUCTION DU TAUX DES REDEVANCES <input type="checkbox"/> requise pour la 1ère fois <input type="checkbox"/> requise antérieurement au dépôt : joindre copie de la décision d'admission											
6 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE <table border="1"> <thead> <tr> <th>pays d'origine</th> <th>numéro</th> <th>date de dépôt</th> <th>nature de la demande</th> </tr> </thead> <tbody> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> </tbody> </table>				pays d'origine	numéro	date de dépôt	nature de la demande				
pays d'origine	numéro	date de dépôt	nature de la demande								
7 DIVISIONS antérieures à la présente demande n° date n° date											
8 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (nom et qualité du signataire - n° d'inscription) BALLOT Paul 92-1009 CABINET BALLOT-SCHMIT 		SIGNATURE DU PRÉPOSÉ À LA RÉCEPTION SIGNATURE APRES ENREGISTREMENT DE LA DEMANDE À L'INPI 									

DIVISION ADMINISTRATIVE DES BREVETS

26bis, rue de Saint-Petersbourg

75800 Paris Cédex 08

Tél. : (1) 42 93 52 52 Télécopie : (1) 42 93 59 30

N° D'ENREGISTREMENT NATIONAL

990292h

TITRE DE L'INVENTION :

**PROCEDE DE SURVEILLANCE DU DEROULEMENT D'UN PROGRAMME,
DISPOSITIF DE SURVEILLANCE DE PROGRAMME, DISPOSITIF D'EXECUTION
DE PROGRAMME AVEC SURVEILLANCE DE PROGRAMME ET DISPOSITIF
PROGRAMME PERMETTANT LA SURVEILLANCE DE SON PROGRAMME.**

LE (S) SOUSSIGNÉ (S)

Cabinet BALLOT SCHMIT
7, rue Le Sueur
75116 PARIS
FRANCE

DÉSIGNE (NT) EN TANT QU'INVENTEUR (S) (indiquer nom, prénoms, adresse et souligner le nom patronymique) :

NACCACHE David
GIRARD Pierre
ROUSSEAU Ludovic

domicilié (s) au :

Cabinet BALLOT SCHMIT
7, rue Le Sueur
75116 PARIS
FRANCE

NOTA : A titre exceptionnel, le nom de l'inventeur peut être suivi de celui de la société à laquelle il appartient (société d'appartenance) lorsque celle-ci est différente de la société déposante ou titulaire.

Date et signature (s) du (des) demandeur (s) ou du mandataire

Paul Ballot le 9 mars 1999

Paul BALLOT - 92-1009
Cabinet BALLOT SCHMIT

DOCUMENT COMPORTANT DES MODIFICATIONS

PAGE(S) DE LA DESCRIPTION OU DES REVENDECATIONS OU PLANCHE(S) DE DESSIN			R.M.*	DATE DE LA CORRESPONDANCE	TAMPON DATEUR DU CORRECTEUR
Modifiée(s)	Supprimée(s)	Ajoutée(s)			
23			R.M.	20.09.99	GY 27.10.99

Un changement apporté à la rédaction des revendications d'origine, sauf si celui-ci découle des dispositions de l'article R.612-36 du code de la Propriété Intellectuelle, est signalé par la mention «R.M.» (revendications modifiées).

**Procédé de surveillance du déroulement d'un programme,
dispositif de surveillance de programme, dispositif d'exécution
de programme avec surveillance de programme et dispositif programmé
permettant la surveillance de son programme**

La présente invention concerne le domaine de la sécurité des programmes informatiques, et plus particulièrement un procédé et un dispositif de détection d'un déroulement inadmissible dans l'exécution d'un programme informatique, celui-ci pouvant être dans un langage de bas niveau ou de haut niveau.

5 Dans un programme en langage de bas niveau, les commandes sont formulées selon une structure très proche de celle des instructions effectivement exécutées par la partie processeur de l'ordinateur. Le programme nécessite seulement d'être compilé avant de pouvoir être exécuté. Les langages de bas niveau, connues sous l'appellation de code machine, sont utilisés notamment pour la programmation de
10 microprocesseurs ou de microcontrôleurs. Les microcontrôleurs sont des processeurs qui ne peuvent exécuter qu'un nombre réduit d'instructions spécifiques. Ils sont utilisés notamment pour équiper des cartes à puce (cartes bancaires, téléphoniques, d'accès à des services, etc.) et pour piloter des appareils industriels ou domestiques.

 Dans un programme en langage de haut niveau, les commandes ont une
15 structure plus proche du langage naturel, mais en revanche plus éloignée de celle utilisée par le processeur. Les commandes écrites dans de tels langages doivent d'abord être interprétées, c'est-à-dire converties en commandes de code machine, avant de pouvoir être ensuite mises sous forme d'instructions en vu de leur exécution par le processeur.

20 Ainsi, tout programme informatique donne lieu à une suite d'instructions adaptées au processeur, microprocesseur ou microcontrôleur auquel il est destiné.

 De manière classique, les instructions d'un programme sont exécutées par un processeur selon une séquence régie par un compteur d'instructions, comme il sera brièvement décrit par référence à la figure 1.

25 Les instructions compilées d'un programme sont chargées en blocs d'instructions successives Inst.1, Inst. 2, Inst. 3,, Inst. (où n est un entier) sous forme de codes ou de microcodes dans un registre d'instructions 2. Chaque instruction est identifiée par une adresse spécifique dans ce registre 2. Dans l'exemple, on désigne les adresses des instructions Inst.1, Inst. 2, Inst. 3,, Inst. n

respectivement Ad.1, Ad.2, Ad.3, ..., Ad.n. Les instructions sont lues du registre d'instructions 2 et chargées successivement dans le processeur 4 pour y être exécutées sous la commande d'un compteur d'instructions 6, lui-même contrôlé par le processeur 4. A cette fin, le compteur d'instructions 6 comporte un pointeur d'adresses 8 qui désigne l'adresse Ad.1, ..., Ad.n du registre 2 à partir de laquelle doit être lue l'instruction à charger dans le processeur 4 durant l'exécution d'une séquence d'instructions. La position du pointeur 8 vis-à-vis des adresses du registre d'instructions 2 évolue donc au fur et à mesure de l'exécution des instructions.

Dans l'exemple représenté à la figure 1, les instructions Inst.1, Inst. 2, Inst. 3, ..., Inst.n dans le registre 2 doivent être exécutées successivement de la première instruction Inst. 1 à la n^{ième} instruction Inst.n, c'est-à-dire de façon linéaire. Ainsi, le pointeur 8 du compteur d'instructions 6 désigne initialement l'adresse Ad.1 du registre 2, et les données 10-1 de l'instruction correspondante Inst.1 sont chargées dans le processeur 4. Lorsque le processeur 4 ordonne au compteur d'instructions 6 de fournir la prochaine instruction (dans ce cas Inst.2), celui-ci incrémente la position du pointeur 8 d'une unité d'évolution d'adresse afin de désigner l'adresse Ad.2. Ce processus se répète et se termine lorsque le pointeur 8 désigne l'adresse Ad.n afin de charger les données 10-n de la dernière instruction Inst.n (pointillées).

Une série d'instructions exécutées de façon linéaire ne comporte pas de "sauts" qui échappent de la progression séquentielle du pointeur 8 vis-à-vis des adresses successives. Ceci est le cas, par exemple, avec une séquence d'instructions en microcodes comme :

```

lda    t
txa
mul
bset   3,t
sta    n

```

Autrement dit, cette séquence s'exécutera de façon linéaire, le compteur d'instructions 6 étant incrémenté d'une unité d'évolution d'adresse lors du passage d'une instruction à l'autre.

Cependant, il est courant que l'exécution d'un programme nécessite des sauts vers des instructions en dehors de la séquence linéaire des instructions telle qu'elle se présente dans le registre 2. De tels sauts peuvent résulter d'une instruction de

chargement de données situées dans une adresse en dehors de la séquence, ou d'une instruction, dite de branchement, d'exécution conditionnelle de la commande suivante.

Une instruction qui provoque un saut nécessite, au niveau du compteur
 5 d'instructions 6, de déterminer l'adresse dans le registre 2 de la prochaine instruction suivant ce saut et de positionner le pointeur 8 à cette adresse afin que l'instruction ou la donnée qui s'y trouve soit chargée dans le processeur 4.

A titre d'exemple, la séquence :

```

lda
10 txa
bra label ; saut
mul
bset 3,t
sta n
15 label rts

```

provoquera le chargement d'une nouvelle valeur dans le compteur d'instructions 6 à l'endroit du code correspondant au commentaire "saut".

Le fait pouvoir ainsi effectuer sous commande des sauts vers des adresses en dehors d'une suite séquentielle d'adresses peut malheureusement donner lieu à un
 20 déroulement inadmissible du programme. Un tel déroulement inadmissible du programme peut résulter d'un dysfonctionnement accidentel du dispositif programmé. Mais il peut aussi provenir d'une action malveillante visant à détourner le fonctionnement du dispositif programmé de sa fonction voulue. Par exemple, dans le cas d'une carte à puce, des modifications de programmation du microcontrôleur
 25 par la création ou la modification de sauts et/ou de branchements peuvent permettre de charger des données erronées (augmentation de crédit autorisé avec une carte bancaire ou téléphonique, fausse autorisation d'accès à certains services, etc.) ou de récupérer des données confidentielles stockées en mémoire (code d'accès, informations personnelles concernant le titulaire de la carte, etc.).

30 En effet, même lorsque les programmes sont enregistrés de manière figée sur une puce semiconductrice, il est possible avec les techniques actuelles de sondage et de test de composants de créer des contacts de sondage sur la surface de la puce (ou même dans les couches basses de celle-ci) à l'aide de stations de travail à focalisation

de faisceaux d'ions (aussi connues sous le terme anglo-saxon de "Focus Ion Beams", ou "FIBs").

Une fois créés, ces points de sondage permettent le dépôt de pointes de sondage à l'aide d'établis spécialisés (aussi connu sous le terme anglo-saxon de "probe stations") permettant la lecture en continue d'un bit (et le suivi de son évolution au cours du temps) ou une modification externe de sa valeur.

En particulier, le dépôt de contacts sur le registre du compteur d'instructions 6 permet de charger de façon externe la valeur du registre en cours d'exécution et de provoquer un branchement non-prévu par le concepteur du programme. Comme 10 expliqué plus haut, un tel saut peut avoir, bien entendu, des conséquences néfastes sur la sécurité de l'applicatif et aboutir, par exemple, à la divulgation de données secrètes en effectuant des calculs incomplets.

Il existe également des manières plus rudimentaires, mais moins sûres, de provoquer de tels dysfonctionnements dans le déroulement du programme. Un 15 exemple en est donné dans un article intitulé "Tamper-resistance, a cautionary note" par R.Anderson. Une autre technique consiste à exploiter des fautes de calcul provoquées délibérément afin d'extraire d'une carte à puce des données telles que les clés secrètes. Cette technique est décrite dans l'article intitulé "On the Importance of Checking Computations" de Boneh, DeMillo et Lipton; Bellcore Report, publié le 31 20 octobre 1996.

Bien entendu, le même phénomène peut également se produire lorsque le programme attaqué est interprété au lieu d'être compilé. Ainsi, une application en langage Java ou Basic peut être détournée de son usage légitime si l'attaquant arrive, par exemple, à provoquer un changement dans le pointeur de programme de 25 l'interpréter indiquant l'instruction courante à interpréter.

Or, les systèmes informatiques actuels ne sont pas spécifiquement conçus afin d'interdire les branchements incontrôlés à l'intérieur d'un code. Bien au contraire, les langages d'assemblage ont été spécifiquement conçus afin de permettre au programmeur le maximum de liberté. A titre d'exemple, en langage C, il est possible 30 de sauter à l'intérieur du code d'une fonction en utilisant l'exécution indexée par le pointeur.

Au vu de ces problèmes de déroulement de programme de manière inadmissible, qu'ils aient pour cause un dysfonctionnement intempestif ou une

volonté de détourner le programme de son utilisation prévue, l'invention propose un procédé de surveillance du déroulement de l'exécution d'une suite d'instructions d'un programme informatique, consistant à analyser la séquence des instructions transmises vers le processeur destiné à exécuter le programme surveillé et à vérifier le résultat de cette analyse avec des données de référence enregistrées avec ledit programme.

Ainsi, la présente invention permet de vérifier que toutes les instructions comprises dans l'ensemble d'instructions considéré ont bien été transmises vers le processeur en vu de leur exécution. Il est à supposer que si tel est le cas, les instructions lues ainsi auront aussi été exécutées.

Les données de référence peuvent être par exemple une valeur préétablie de manière à correspondre au résultat de l'analyse réalisée lors du procédé de surveillance seulement si toutes les instructions de la séquence d'instructions ont été effectivement analysées lors du déroulement du programme.

De préférence, l'étape d'analyse comprend les sous-étapes d'extraction d'une donnée de chaque instruction transmise vers le processeur et de calcul prédéterminé sur chaque donnée ainsi extraite, et l'étape de vérification comporte la comparaison du résultat de l'analyse avec les données de référence.

Avantageusement, l'étape de vérification s'effectue par une comparaison câblée d'une valeur contenue dans un registre associé aux moyens de surveillance avec la valeur de référence, celle-ci pouvant être inscrite dans le programme de manière câblée, fixée (par exemple dans une mémoire figée type ROM) une fois pour toutes lors du masquage du code qui constitue le programme surveillé.

De préférence, la vérification est provoquée par une instruction placée à un emplacement prédéterminé dans le programme, cette instruction contenant les données de référence précitées.

Avantageusement, lorsque les instructions de l'ensemble d'instructions à surveiller se présentent sous la forme d'une valeur, hexadécimal ou décimal, on traite ces instructions en tant que simples valeur numériques lors de l'analyse précitée.

Le procédé global de surveillance de l'exécution d'une séquence d'instructions d'un programme informatique peut ainsi comprendre les étapes suivantes :

- lors de la préparation du programme:

- incorporer, à au moins un emplacement prédéterminé d'une séquence d'instructions du programme, une valeur de référence établie selon une règle donnée appliquée sur des données identifiables dans chaque instruction à surveiller, et
- 5 - lors de l'exécution de la partie du programme à surveiller:
 - obtenir lesdites données identifiables dans chaque instruction exécutée,
 - appliquer ladite règle donnée sur lesdites données identifiables ainsi obtenues pour établir une valeur de vérification, et
 - 10 - vérifier que cette valeur de vérification correspond effectivement à la valeur de référence enregistrée avec le programme.

Dans un mode de réalisation préféré de l'invention, on prévoit d'interrompre le déroulement du programme lorsque qu'il est détecté que la valeur de vérification ne correspond à la valeur de référence. Cette interruption peut être assortie d'une
15 action d'invalidation pour usage futur du dispositif comprenant le programme informatique surveillé si la non correspondance entre la valeur de vérification et la valeur de référence est détectée un nombre prédéterminé de fois.

Avantageusement, l'ensemble d'instructions à surveiller ne comporte pas de sauts dans son déroulement prévu, de sorte que l'on s'attend à ce que toutes les
20 instructions qu'il comporte soient exécutées dans tous les cas de figure envisagés.

Lorsque le programme ou la portion de programme à surveiller prévoit au moins un saut, il est possible d'appliquer le procédé de surveillance séparément sur des ensembles d'instructions qui ne comportent pas de sauts.

Dans le cas d'une instruction donnant lieu à au moins un saut dépendant des
25 données manipulées, c'est-à-dire un branchement conditionnel, on peut mettre en œuvre le procédé de surveillance séparément pour un ensemble d'instructions dépourvu de sauts et qui précède le saut, et pour au moins un ensemble d'instructions dépourvu de sauts qui succède à ce saut.

Dans ce cas, on peut envisager que pour un ensemble d'instructions précédant
30 un saut, on intègre à cet ensemble l'instruction qui commande le saut (cette instruction étant normalement la dernière avant un branchement) aux fins de l'analyse visant à obtenir la valeur de vérification de cet ensemble d'instructions, et

on vérifie ainsi le bon déroulement de cet ensemble d'instructions avant d'exécuter l'instruction de saut.

Avantageusement, on efface la valeur de vérification obtenue lors d'une précédente mise en œuvre du procédé à chaque nouvelle mise en œuvre du procédé.

- 5 Cette disposition permet de gérer aisément la surveillance de différents ensembles d'instructions d'un programme, tels que ceux séparés par des sauts. Elle permet notamment de mettre en œuvre le procédé avec les mêmes conditions initiales de calcul de valeur de vérification pour les différents ensembles séparés par des sauts.

- 10 A chaque nouvelle mise en œuvre du procédé, la valeur de vérification peut être effacée par une simple remise à zéro. Cette valeur peut également être remplacée par une autre valeur initiale prédéterminée. Ces opérations de remise à zéro ou d'initialisation peuvent être activées par le logiciel protégé lui-même.

- 15 Avantageusement, la valeur de vérification est obtenue en tant que dernière valeur d'une suite de valeurs que l'on fait évoluer successivement avec l'analyse de chacune des instructions considérées de l'ensemble d'instructions. Cette approche permet de contenir un état interne du déroulement du procédé de surveillance et de suivre son évolution.

- 20 De préférence, le mode d'analyse permettant cette évolution de la valeur de vérification consiste à calculer, pour chaque instruction considérée succédant à une instruction précédente, le résultat d'une opération sur à la fois une valeur extraite de l'instruction considérée et le résultat obtenu par la même opération effectuée sur cette instruction précédente. Pour le calcul lié à une première instruction à vérifier, on peut appliquer une opération sur à la fois la donnée extraite de cette première instruction et une valeur prédéterminée (qui peut correspondre alors à la valeur de ré-
25 initialisation ou de remise à zéro précitée), celle-ci faisant office de valeur "graine" en l'absence résultat d'opération précédente.

- 30 De cette manière, il est possible d'obtenir la valeur de vérification correcte en utilisant un algorithme récursif applicable de la même manière pour les données extraites de chacune des instructions considérées. Qui plus est, l'opération de calcul peut être choisie aisément de manière à ce qu'une valeur de vérification correcte soit obtenue seulement si d'une part les données de toutes les instructions ont été considérées lors du calcul, et d'autre part elles ont été considérées dans l'ordre prévu.

L'opération de calcul peut consister à appliquer une fonction de hachage, selon une technique connue en elle-même dans le domaine du chiffrement de données, tel que la fonction de hachage SHA-1 établie par norme fédérale de hachage. On peut réaliser dans ce cas l'évolution interne précitée du déroulement du
5 procédé de surveillance en hachant cryptographiquement l'ensemble des codes d'opération (considérés en tant que valeurs numériques) et des adresses exécutés depuis la dernière initialisation effectuée.

En variante, il est possible de faire évoluer la valeur de vérification en effectuant un calcul de redondance non nécessairement cryptographique sur l'ensemble des
10 codes d'opération et des adresses exécutées depuis la dernière initialisation effectuée. A titre d'exemple, on peut utiliser des algorithmes du type CRC (cyclic redundancy check en anglais).

Il est possible avec l'invention d'obtenir la valeur de comparaison par calcul de valeurs intermédiaires au fur et à mesure que les données comprises dans les
15 instructions respectives sont obtenues au cours de l'exécution de celles-ci. Avec cette approche, il n'est pas nécessaire de sauvegarder chaque valeur extraite des instructions de l'ensemble d'instructions considéré. En effet, à l'issue d'un calcul de valeur intermédiaire, seule compte cette valeur intermédiaire pour calculer la prochaine valeur intermédiaire (ou la valeur ultime qui correspond à la valeur de
20 vérification), et la donnée qui a permis de l'engendrer n'est plus à prendre en considération. Cette disposition permet d'économiser de l'espace mémoire au niveau des moyens de mise en œuvre de l'invention.

En variante, il est possible de sauvegarder chaque donnée comprise dans les instructions de l'ensemble d'instructions considéré au fur et à mesure qu'elles sont
25 exécutées et de n'effectuer le calcul de la valeur de vérification seulement au moment nécessaire, par exemple lors de l'étape de vérification.

L'invention concerne aussi un dispositif de surveillance du déroulement de l'exécution d'une suite d'instructions d'un programme informatique, caractérisé en ce qu'il comporte des moyens pour analyser la séquence des instructions transmises vers
30 le processeur destiné à exécuter le programme surveillé et des moyens pour vérifier le résultat de cette analyse avec des données de référence enregistrées avec ledit programme.

Le dispositif de surveillance conforme à la présente invention comporte avantageusement un registre permettant d'enregistrer des résultats intermédiaires dans le calcul de la valeur de vérification. Ce registre peut être adapté pour ne retenir seulement le dernier résultat intermédiaire en cours.

5 Il peut être prévu pour permettre l'enregistrement d'une valeur prédéterminée ou une remise à zéro sous la commande du programme en cours d'exécution. De cette manière, le programme peut commander une condition initiale au niveau du contenu du registre à chaque nouvelle mise en œuvre du procédé de surveillance, celle-ci intervenant par exemple après un saut dans le programme.

10 Le dispositif de surveillance peut être intégré dans un dispositif d'exécution de programme à surveiller ou dans un dispositif programmé qui contient le programme à surveiller.

L'invention concerne également un dispositif d'exécution de programme, par exemple un ordinateur, un appareil à microprocesseur ou à microcontrôleur tel qu'un
15 lecteur de carte à puce ou de programme enregistré sur une carte au format PCMCIA, destiné à exécuter une suite d'instructions d'un programme informatique, caractérisé en ce qu'il comporte des moyens pour analyser la séquence des instructions transmises pour exécution et des moyens pour vérifier que le résultat de cette analyse correspond avec des données de référence enregistrées avec le programme.

20 L'invention concerne aussi un dispositif programmé destiné à fonctionner avec le dispositif d'exécution de programme précité et comportant une suite d'instructions, caractérisé en ce qu'il comporte en outre des données de référence préétablies en fonction de données contenues dans lesdites instructions et destinées à permettre une vérification de la séquence des instructions analysées par le dispositif
25 d'exécution de programme précité.

Le dispositif programmé, par exemple une carte à puce ou un organe de contrôle de mécanisme, tel qu'un système de freinage ABS, peut intégrer le programme à surveiller dans une mémoire figée du type ROM.

Les données de référence sont avantageusement enregistrées sous la forme de
30 valeur(s) pré-câblée(s) fixée(s) dans la mémoire une fois pour toutes lors du masquage du code.

La présente invention concerne aussi un dispositif de programmation d'un dispositif d'exécution d'un programme destiné à fonctionner en association avec le

dispositif programmé précité, caractérisé en ce qu'il comprend des moyens pour inscrire, à au moins un emplacement prédéterminé d'une séquence d'instructions du programme, une valeur de référence calculée selon un mode préétabli à partir de données comprises dans chaque instruction d'un ensemble d'instructions dont on
5 souhaite surveiller l'exécution.

Enfin, l'invention concerne aussi une machine virtuelle ou interpréteur interprétant un code critique, caractérisé en ce qu'il met en œuvre le procédé de surveillance précité pour l'exécution de ce code critique.

Les dispositifs précités de surveillance, d'exécution de programme, ou de
10 programmation ou encore les dispositifs équipés de tels programmes peuvent être équipés de tous les moyens nécessaires pour réaliser les différents aspects optionnels possibles du procédé de surveillance précité.

A titre d'exemple, on peut envisager dans une application liée à une carte à puce, d'ajouter à un microprocesseur qui exécute un programme un composant
15 matériel additionnel servant d'unité de surveillance. Le rôle de cette unité est de surveiller que des sauts non-prévus par le concepteur du logiciel ne puissent avoir lieu en cours d'exécution. Dans cet exemple, l'unité de surveillance pourra être composée de registres dont le contenu constitue à tout moment l'état interne de l'unité de surveillance. Une entrée spécifique de l'unité de surveillance permet sa remise à
20 zéro (RAZ), typiquement en effaçant le contenu de l'unité de surveillance. Cette fonction peut être activée à tout moment par le logiciel exécuté et peut, par exemple, se réaliser par l'ajout d'un nouveau code opération en assembleur (par exemple "clr us") ou en manipulant un bit donnée dans la mémoire du composant protégé (par exemple : setb 3, service).

25 Dans cet exemple d'application, l'unité de surveillance compare son état interne avec une chaîne donnée fournie par le logiciel protégé. Cela peut par exemple être réalisé en recopiant à l'intérieur de l'unité de surveillance (à l'aide d'une boucle "lda-sta") la valeur à laquelle le logiciel souhaite comparer l'état interne. Une fois la recopie de la valeur terminée, l'unité de surveillance la compare à son état
30 interne et adopte le comportement suivant: si l'état de l'unité de surveillance est égal à la valeur présentée par le logiciel protégé, reprendre l'exécution normalement, sinon l'exécution du programme est arrêtée (forçant l'utilisateur à faire une remise à zéro de la carte), éventuellement en ratifiant au préalable un compteur de fausses

exécutions dans une mémoire non volatile du type EEPROM ayant pour effet le blocage définitif de la carte si sa valeur dépasse une limite raisonnable (par exemple 4).

5 L'unité de surveillance peut garder de façon permanente un haché cryptographique des codes instructions et des adresses exécutés depuis sa dernière remise à zéro.

Le mécanisme de surveillance peut être adapté à l'interprétation de code dans une machine virtuelle (du "byte code" Java, par exemple). Le compilateur peut calculer la valeur du haché d'une portion de byte code, l'intégrer dans un attribut d'une structure connue sous le terme anglo-saxon de "class file" produit et ajouter au
10 byte code généré des codes connus sous le terme anglo-saxon de "opcodes" supplémentaires correspondant à la remise à zéro de l'unité de surveillance et à l'appel de la fonction de vérification. La machine virtuelle tiendra lieu d'unité de surveillance et lorsqu'elle rencontrera l'opcode de vérification, vérifiera la valeur du
15 haché courant par rapport à la valeur du haché théorique contenu dans le class file.

L'invention sera mieux comprise et les avantages et caractéristiques ressortiront plus clairement à la lecture de la description suivante d'un mode de réalisation préféré, donnée purement à titre d'exemple, par référence aux dessins annexés, dans lesquels :

- 20 - la figure 1, déjà présentée, est un schéma bloc simplifié visant à expliquer le rôle d'un compteur d'instructions dans le déroulement de l'exécution d'un programme,
- la figure 2 est un schéma bloc simplifié d'un dispositif d'exécution de programme visant à expliquer le principe de fonctionnement d'une unité de surveillance conformément à un premier mode de réalisation de l'invention,
- 25 - la figure 3 est un organigramme du procédé de surveillance selon l'invention,
- la figure 4 est un organigramme d'une variante du procédé de surveillance selon l'invention,
- la figure 5 est un schéma bloc simplifié d'un dispositif d'exécution de
30 programme visant à expliquer le principe de fonctionnement d'une unité de surveillance conformément à un deuxième mode de réalisation de l'invention,
- la figure 6 est un organigramme du procédé de surveillance selon l'invention, adapté au deuxième mode de réalisation, et

- la figure 7 représente de manière schématisée des ensembles d'instructions d'un programme avec branchement contenant en outre des instructions spécifiques au procédé de surveillance.

Le principe de l'invention sera expliqué par référence au schéma bloc de la figure 2, dans lequel les blocs ayant un rôle analogue à ceux de la figure 1 portent les mêmes références et ne seront pas décrits à nouveau par souci de concision.

La figure 2 représente les éléments de base d'un dispositif d'exécution de programme 20 au sens large. Il peut s'agir d'un ordinateur destiné à exécuter un programme dans un langage de haute niveau, d'un microprocesseur ou d'un microcontrôleur, ces derniers fonctionnant à partir de programmes en langage de bas niveau. A titre d'exemple, le dispositif d'exécution 20 peut être un lecteur de carte à puce destinée à gérer des transactions bancaires, téléphoniques, ou d'autres services. Le programme à vérifier est alors contenu matériellement dans la carte à puce.

Afin de concrétiser la description qui suit, on supposera que le dispositif d'exécution de programme 20 est basé sur un processeur 4 de type microcontrôleur.

Le processeur 4 exécute une portion de programme stockée dans un registre d'instructions 2 sous forme de microcodes. La partie opérationnelle de cette portion de programme comprend une séquence de n instructions (où n est un entier supérieur à 1) désignées respectivement Inst.1, Inst.2, Inst.3, ..., Inst. n . Les microcodes qui constituent les instructions se présentent sous la forme de valeurs numériques, qui peuvent être décimales ou hexadécimales.

Une telle valeur peut ainsi être considérée de deux façons distinctes, chacune porteuse d'une donnée : premièrement en tant qu'instruction qu'elle représente pour le processeur (dans lequel cas elle sera désignée "valeur code"), et deuxièmement en tant que simple valeur numérique susceptible de traitement arithmétique (dans lequel cas elle sera désignée "valeur numérique" Vinst.). Par exemple, la première instruction Inst.1 est égale à 40. Ce chiffre est un code qui correspond à une instruction reconnue par le processeur, mais il possède la même structure binaire que la valeur numérique 40.

Aucune des instructions Inst.1, Inst.2, Inst.3, ..., Inst. n ordonne un saut vers une autre instruction en dehors de la séquence linéaire d'exécution des instructions. Ainsi, le déroulement normal et prévu de cette portion de programme requiert obligatoirement l'exécution de chacune des instructions en succession, en

commençant par l'instruction Inst.1 et en terminant par l'instruction Inst.n. De la sorte, le compteur d'instructions 6 (déjà décrit) positionnera son pointeur 8 successivement à l'adresse de chacune des instructions Inst.1 à Inst.n dans le registre d'instruction 2 au fur et à mesure qu'elles doivent être chargées dans le processeur 4.

5 Conformément à la présente invention, le dispositif d'exécution de programme 20 comporte une unité de surveillance 22 qui permet de vérifier que chacune des instructions Inst.1 à Inst.n a bien été chargée vers le processeur 4 en vu de leur exécution. Elle est fonctionnellement reliée entre le registre d'instructions 2 et le processeur 4. Ainsi, toutes les instructions lues du registre d'instructions 2
10 transitent à travers l'unité de surveillance 22 avant de parvenir au processeur 4.

Dans cet exemple, l'unité de surveillance 22 est décrite comme étant intégrée au dispositif d'exécution de programme 20. Toutefois, l'unité de surveillance 22 peut tout aussi bien être intégrée avec le dispositif qui comporte le programme à surveiller, en étant par exemple incorporée dans une carte à puce dont le programme
15 qu'elle contient en mémoire est à surveiller, sans que cela change les principes qui seront décrits ci-après.

Comme il sera expliqué de manière plus détaillée ci-après, l'unité de surveillance 22 comporte un registre 24 destiné à stocker provisoirement une donnée comprise dans une instruction Inst.1, Inst.2, Inst.3, ..., Inst.n et un calculateur 26
20 destiné à exécuter une opération sur cette donnée.

La mise en œuvre de l'unité de surveillance demande l'adjonction de deux nouvelles instructions aux n instructions Inst.1, Inst.2, Inst.3, ..., Inst.n du programme: une première instruction de surveillance Inst.0 placée avant la première instruction Inst.1 du programme et une seconde instruction de surveillance Inst.n+1
25 placée à la suite de la dernière instruction Inst.n du programme.

Lors du déroulement de l'exécution des n instructions du programme, le compteur d'instructions 6 est initialement commandé pour positionner son pointeur 8 à l'adresse de la première instruction de surveillance Inst.0. Cette instruction commande à l'unité de surveillance d'initialiser une valeur de hachage VH contenue
30 dans son registre 24. Dans l'exemple, l'instruction Inst.0 commande simplement au registre 24 mettre la valeur $VH = 0$. Elle n'est pas transmise vers le processeur 4.

Ensuite, le dispositif d'exécution de programme 20 passe en phase d'exécution des instructions Inst.1, Inst.2, Inst.3, ..., Inst.n du programme proprement dit.

Chaque instruction lue à partir du registre d'instructions 2 est transmise en premier lieu à l'unité de surveillance 22 où elle est considérée comme valeur numérique.

La valeur numérique de chaque instruction est soumise par le calculateur 26 à un algorithme de hachage, tel le hachage SHA-1 précisé par la norme fédérale de hachage. Le résultat VH_i de l'opération de hachage liée à une instruction $Inst.i$ (où i est un entier de 1 à n) est inscrite dans le registre 24.

Cette valeur VH_i sert de base pour l'opération de hachage avec l'instruction suivante $Inst.i+1$. Le résultat de hachage VH_{i+1} ainsi obtenu pour l'instruction $Inst.i+1$ est alors inscrit en lieu et place du résultat de hachage VH_i obtenu précédemment.

Cette procédure est poursuivie pour chacune des instructions $Inst.1$, $Inst.2$, $Inst.3$, ..., $Inst.n$ qui transitent par l'unité de surveillance 22.

Lorsque la dernière instruction $Inst.n$ est exécutée, la seconde instruction de surveillance $Inst.n+1$ est chargée dans l'unité de surveillance 22. Cette instruction comporte deux composantes : une valeur de référence $V_{réf}$ et une commande, destinée au calculateur 26, de comparaison de cette valeur de référence $V_{réf}$ avec la valeur du dernier résultat de hachage inscrite dans le registre 24. Cette dernière valeur correspond donc au résultat du hachage VH_n obtenu à partir de la valeur numérique de l'instruction $Inst.n$ (égale à 36 dans la figure) et du résultat de hachage VH_{n-1} obtenue pour l'instruction $Inst.n-1$ précédente.

Ainsi, en réponse à la seconde instruction de surveillance $Inst.n+1$, le calculateur 26 compare la valeur VH_n dans le registre 22 avec la valeur de référence $V_{réf}$ spécifiée dans cette instruction de surveillance.

La valeur de référence $V_{réf}$ est déterminée lors de la préparation du programme enregistré afin de correspondre à la valeur attendue VH_n pour le résultat des hachages successifs des valeurs des instructions $Inst.1$, $Inst.2$, $Inst.3$, ..., $Inst.n$. Cette valeur $V_{réf}$ peut être calculée d'avance en utilisant la même procédure de hachage successif des instructions $Inst.1$, $Inst.2$, $Inst.3$, ..., $Inst.n$ qu'utilisée par l'unité de surveillance 22.

De préférence, la valeur $V_{réf}$ est câblée dans une mémoire figée afin de ne pas pouvoir être modifiée par un acte de malveillance.

Si l'unité de surveillance 22 constate, en exécutant l'instruction du surveillance $Inst.n+1$, qu'il y a identité entre les valeurs $V_{réf}$ et VH_n précitées, il est

conclu que toutes les instructions Inst.1, Inst.2, Inst.3, ..., Inst.n. ont bien été transmises vers le processeur 4 en vu de leur exécution.

Si, au contraire, l'unité de surveillance 22 constate qu'il n'y a pas identité entre les valeurs $V_{réf}$ et V_{Hn} , il est conclu que ou bien toutes les instructions Inst.1, Inst.2, Inst.3, ..., Inst.n. n'ont pas été reçues et transmises par l'unité de surveillance, ou bien elles ne l'ont pas été dans l'ordre séquentiel prévu. Dans ce cas, on peut prévoir une action visant alerter l'utilisateur ou le titulaire du programme, ou d'empêcher le programme de se poursuivre. Dans l'exemple, une telle action est transmise de l'unité de surveillance 22 vers le processeur 4 sous forme d'une commande d'interruption du programme Int..

Le procédé de surveillance tel que réalisé par les moyens de la figure 2 sera maintenant décrit par référence à l'organigramme représenté à la figure 3. On suppose que le programme ou la partie de programme à surveiller a été préparé correctement pour le procédé de surveillance par l'incorporation des première et seconde instructions de surveillance respectivement au début et à la fin.

Au stade initial, l'unité de surveillance 22 se positionne sur une routine de début de surveillance 30, dont la première étape 32 est l'attente de la première instruction de surveillance (Inst.0).

Lorsque la première instruction de surveillance Inst.0 est reçue, l'unité de surveillance 22 effectue une étape 34 d'initialisation (par remise à zéro) d'un compteur d'instructions et du registre 24. La remise à zéro du registre 24 est une manière de placer une valeur "graine" dans ce registre pour démarrer une séquence d'opérations de hachage, comme il sera expliqué plus loin. Ces opérations peuvent être commandées directement par la première instruction de surveillance ou être simplement déclenchée par celle-ci à partir d'une routine associée à l'unité de surveillance 22.

Dans ce premier cas, la remise à zéro peut se réaliser par l'ajout d'un nouveau code opération en assembleur (par exemple "clr us"), ou en manipulant un bit donné dans la mémoire du dispositif d'exécution de programme 20. Une telle commande peut être "setb 3, service".

Après l'étape d'initialisation, l'unité de surveillance 22 incrémente le compteur d'instructions d'une unité (positionnant alors ce compteur à $n=1$) (étape 36).

Ensuite, la première instruction Inst.1 du programme ou de la partie de programme sous surveillance est lue du registre d'instructions 2 (étape 38). Comme expliqué plus haut, cette instruction est considérée par l'unité de surveillance 22 purement en tant que valeur numérique permettant des opérations arithmétiques.

5 Dans l'exemple de la figure 2, cette valeur est 40.

La valeur numérique de cette première instruction Inst.1 est alors soumise à une opération de hachage avec la valeur contenue dans le registre 24 (étape 40). Dans le cas de la première instruction, cette dernière valeur est la valeur d'initialisation, c'est-à-dire 0.

10 L'opération de hachage, bien connue en elle-même, consiste ici à faire agir sur la valeur de l'instruction n considérée un opérateur mathématique $f(VH_{n-1}, V_{inst.n})$, où VH_{n-1} est le résultat d'une opération de hachage précédente (ou la valeur d'initialisation dans le cas de la première instruction) enregistré dans le registre 24 et $V_{inst.n}$ est la valeur numérique de l'instruction n considérée.

15 Le résultat VH_n de cette opération de hachage est ensuite enregistré dans le registre 24 à la place du résultat VH_{n-1} précédent (étape 42). On note que cette procédure de réactualisation du contenu du registre à chaque opération de hachage permet de garder de façon permanente un haché cryptographique des codes instructions et des adresses exécutés depuis la dernière initialisation.

20 A la fin de cette opération de hachage, l'instruction est transmise vers le processeur 4 en vue de son exécution (étape 44).

Ensuite, l'unité de surveillance 22 détermine si le programme ou la partie de programme à surveiller contient une autre instruction à exécuter (étape 46).

25 Ceci étant le cas, la procédure effectue un rebouclage B1 vers l'étape 36 d'incrémentation de n à $n+1$. La valeur de la prochaine instruction (Inst.2) sera alors lue du registre d'instructions 2 et soumise à l'opération de hachage de la même manière que pour l'instruction Inst.1. Seulement, le hachage s'effectue cette fois avec d'une part la valeur numérique de l'instruction Inst.2 et le résultat obtenu lors de l'opération de hachage précédente, c'est-à-dire la valeur VH_1 (n étant ici égal à 2) qui
30 est alors dans le registre 24.

Les étapes 42 à 46 du procédé s'effectuent de la même manière que pour la première instruction. De la sorte, l'ensemble des étapes 36 à 46 se poursuit en boucle pour chaque instruction Inst.1, Inst.2, Inst.3, ..., Inst. n . lue du registre d'instructions

2, avec le hachage s'opérant, pour une instruction Inst.i (où i est un entier de 1 à n) avec la valeur VH_{ni-1} dans le registre 24 et la valeur $V_{inst.i}$.

Une fois que toutes les instructions Inst.1, Inst.2, Inst.3, ..., Inst.n. ont été ainsi traitées par l'unité de surveillance 22, celle-ci reçoit la seconde instruction de surveillance Inst. n+1, celle-ci suivant la dernière instruction Inst.n du programme ou
5 de la partie de programme surveillé.

Cette seconde instruction de surveillance commande l'unité de surveillance 22 d'extraire la valeur de référence $V_{réf}$ du programme (étape 48) et de comparer le contenu du registre 24 à cette valeur $V_{réf}$ (étape 50). Cette commande peut être
10 réalisée à l'aide d'une boucle "Ida-sta".

On rappelle que, par exécution de boucles successives des étapes 36 à 46, la valeur contenue dans le registre 24 à ce stade est le résultat VH_n du hachage réalisé avec le résultat VH_{n-1} du hachage précédent et de la valeur numérique de l'instruction n (égal à 36 dans l'exemple de la figure 2).

15 La valeur de référence $V_{réf}$ a été préalablement déterminée lors de la préparation du programme en connaissance des opérations de hachage, afin d'être égale à ce que l'unité de surveillance 22 devrait rendre en tant que valeur VH_n si toutes les instructions Inst.1, Inst.2, Inst.3, ..., Inst.n. ont bien été transférées vers le processeur 4.

20 Ainsi, le résultat de la comparaison permet de contrôler le bon déroulement des instructions Inst.1, Inst.2, Inst.3, ..., Inst.n. : si $VH_n = V_{réf}$ (étape 52), on suppose que toutes les instructions ont été effectivement transférées vers le processeur 4. L'opération de surveillance est alors terminée en ce qui concerne le programme ou la portion de programme contenant les instructions Inst.1, Inst.2, Inst.3, ..., Inst.n.

25 La procédure de surveillance retourne alors à la phase de début 30 en attente d'une nouvelle première instruction de surveillance.

Si, au contraire, l'étape de comparaison 50 fait apparaître qu'il n'y a pas identité entre les valeurs comparées ($VH_n \neq V_{réf}$), on suppose que toutes les instructions Inst.1, Inst.2, Inst.3, ..., Inst.n. n'ont pas été transférées vers le
30 processeur 4, ou n'ont pas été transférées dans le bon ordre (étape 54). En effet, le résultat d'une succession d'opérations de hachage dépend de l'ordre dans lequel elles ont été effectuées.

Dans ce cas, l'unité de surveillance 22 commande une action (étape 56), telle que l'interruption du programme et/ou l'enregistrement du fait que le programme ne s'est pas déroulé correctement.

Une variante du procédé de surveillance précité sera maintenant décrite par
 5 référence à l'organigramme de la figure 4. Selon cette variante, au lieu d'effectuer une opération de hachage lors de la réception de chaque nouvelle instruction par l'unité de surveillance 22, on réalise l'ensemble des opérations de hachage seulement après réception de la seconde instruction de surveillance. Dans l'organigramme de la figure 4, les étapes qui sont identiques à celles précédemment décrites par référence à
 10 la figure 3 portent les mêmes références et ne seront pas décrites à nouveau par souci de concision.

Le procédé de surveillance se déroule comme pour le cas précédent en ce qui concerne les étapes 30 à 38 (figures 3 et 4). Après l'étape 38 de lecture de la valeur de l'instruction Vinst.n du registre d'instructions 2, l'unité de surveillance 22 procède
 15 à l'enregistrement de cette valeur (étape 39). Cet enregistrement peut s'effectuer dans un registre interne du calculateur 26, dans une section dédiée du registre 24, dans une mémoire spécifique (non représentée) de l'unité de surveillance 22, ou encore dans une mémoire extérieure à l'unité de surveillance 22, dès lors qu'elle est accessible par celle-ci.

20 Ensuite, l'unité de surveillance 22 procède aux étapes 44 et 46 précédemment décrites. On remarque que l'étape 39 d'enregistrement des valeurs Vinst.n est dans la boucle B1 reliant l'étape 46 à l'étape 36 d'incrément de n par une unité, de sorte que chacune des valeurs Vinst.n est ainsi enregistrée jusqu'à la détection de la seconde instruction de surveillance à l'étape 46.

25 Lorsque cette seconde instruction apparaît, l'unité de surveillance 22 lit la valeur de référence Vréf (étape 48) et effectue à l'étape 49 le hachage selon le même algorithme qu'aux étapes précédemment décrites 40 et 42 de la figure 3, en se basant sur l'ensemble des valeurs Vinst. préalablement enregistrées. La valeur finale de hachage VHn est alors la même que dans le cas du procédé de la figure 3. On notera
 30 qu'il est possible d'intervertir l'ordre des étapes 48 et 49.

Les étapes de comparaison 50 et celles qui suivent sont identiques à celles de la figure 3.

La figure 5 est un schéma bloc simplifié de l'unité de surveillance 22 conformément à un deuxième mode de réalisation de l'invention. Son intégration dans le dispositif d'exécution de programme 20 est la même que pour le premier mode de réalisation décrit par référence aux figures 2 et 3 en ce qui concerne son fonctionnement avec le compteur d'instructions 6, le registre d'instructions 2 et le processeur 4, et ne sera pas répété par souci de concision.

L'unité de surveillance 22 selon le deuxième mode de réalisation se distingue de celle du premier mode de réalisation essentiellement par le fait qu'il comporte en outre une mémoire 60 qui enregistre le nombre de fois que l'exécution d'une série d'instructions Inst.1-Inst.n ne s'est pas déroulée correctement selon les critères expliqués par référence à la figure 3 ou 4.

Dans l'exemple, la mémoire 60 est matérialisée sous forme de mémoire figée (non volatile) à contenu effaçable électriquement (connu communément par l'appellation anglo-saxonne EEPROM).

La mémoire 60 est reliée fonctionnellement au calculateur 26 de manière à ce qu'elle enregistre une valeur de comptage VC incrémentée d'une unité à chaque constat d'une exécution incorrecte dans la série d'instructions surveillée. Cette valeur de comptage VC permet ainsi de détecter le nombre de déroulements incorrects de la série d'instructions et d'agir en conséquence, par exemple en invalidant toute future utilisation du dispositif contenant le programme (par exemple une carte à puce) si ce nombre franchit une valeur seuil.

L'organigramme de la figure 6 donne un exemple d'utilisation de la valeur de comptage VC pour contrôler le dispositif de l'exécution de programme. Cet exemple comporte l'ensemble des étapes 30 à 54 de l'organigramme de la figure 3 ou les étapes analogues de la figure 4.

Lorsque l'unité de surveillance 22 détecte à l'étape 54 une exécution non-prévue des instructions Inst.1 - Inst.n, suite à l'étape de comparaison 52, l'unité de calcul 26 incrémente la valeur de comptage VC dans la mémoire 60, initialement égale à 0, d'une unité (étape 62). Ensuite, elle vérifie si la valeur de comptage VC ainsi incrémentée a atteint une valeur seuil VCseuil prédéterminée (étape 64). Cette valeur seuil VCseuil correspond au nombre de fois qu'il est admis que l'exécution non-prévue des instructions Inst.1-Inst.n puisse survenir dans le dispositif programmé avant de prendre des mesures définitives pour faire face à une telle

défaillance. A titre d'exemple dans le contexte d'une carte à puce, on peut admettre un nombre modéré de telles défaillances (par exemple 3 ou 4) au bénéfice du doute qu'il pourrait s'agir d'une avarie momentanée liée au lecteur (dispositif d'exécution de programme 20), mais qu'au delà de ce nombre, on doit considérer que la carte a été

5 altérée, soit accidentellement, soit par malveillance. Dans ce cas de mise en œuvre, on peut aussi prévoir d'inscrire la valeur VC dans le dispositif programmé (carte) afin de garder un historique de ces exécutions défaillantes lié physiquement au dispositif programme.

Si la valeur de comptage VC est inférieure au seuil VCseuil, l'unité de

10 surveillance 22 établit la commande d'interruption Int. telle que précédemment décrite avec un simple message d'alerte destiné à l'utilisateur et/ou au système d'exploitation (étape 66) et la transmet au processeur 4 (étape 68).

Si, au contraire, la valeur de comptage VC atteint la valeur seuil VCseuil, l'unité de surveillance 22 établit alors la commande d'interruption Int. telle que

15 précédemment décrite avec un ordre d'interdire toute future utilisation du dispositif programmé contenant les instructions exécutées de manière non-prévue (étape 70) et la transmet au processeur (étape 68). Dans ce cas, il ne sera possible de réutiliser ce dispositif seulement après avoir re-programmé la mémoire 60. Dans le cas où cette mémoire 60 est sous forme de EEPROM ou autre mémoire non-volatile, une telle re-

20 programmation est très difficile à réaliser de manière détournée.

On notera que la commande Int. d'interruption du programme assortie d'une transmission de message d'alerte ou de commande d'invalidation d'usage futur peut être exécutée soit au niveau du processeur, soit au niveau de l'unité de surveillance 22.

25 Il sera maintenant décrit par référence à la figure 7 comment l'unité de surveillance 22 conformément à la présente invention peut être mise en œuvre pour la surveillance d'un programme qui prévoit des sauts ou des branchements.

Dans l'exemple de la figure 7, le dispositif d'exécution de programme 20 comporte dans le registre d'instructions 2 un programme ou une partie de programme

30 destiné au processeur 4, constitué de trois ensembles d'instructions :

- un premier ensemble d'instructions Inst.EI1-1 à Inst.EI1-j (où j est un entier > 1), dont la dernière instruction EI1-j est un code qui commande le branchement conditionnel vers l'un ou l'autre des deux autres ensembles qui suivent;

- un deuxième ensemble d'instructions Inst.EI2-1 à Inst.EI2k (où k est un entier > 1) ; l'exécution de la première instruction Inst.EI2-1 de cet ensemble succède à l'exécution de l'instruction de branchement conditionnel EI1-j si la première de deux conditions posées par celle-ci est satisfaite; et

- 5 - un troisième ensemble d'instructions Inst.EI3-1 à Inst.EI3l (où l est un entier > 1); l'exécution de la première instruction Inst.EI3-1 de cet ensemble succède à l'exécution de l'instruction de branchement conditionnel EI1-j si la seconde de deux conditions posées par celle-ci est satisfaite.

Les trois ensembles d'instructions EI1, EI2 et EI3 ne comportent pas de sauts
10 à l'intérieur de leur séquence d'instructions. (Dans le cas du premier ensemble d'instructions, le saut conditionnel à l'instruction EI1-j est à la fin de la séquence.) Ainsi, pour chacun des trois ensembles d'instructions, toutes les instructions sont prévues pour être exécutées séquentiellement à partir de la première.

Lors de la préparation du programme, on ajoute en tête et en queue de chaque
15 ensemble d'instructions EI1, EI2 et EI3 respectivement la première instruction de surveillance et la seconde instruction de surveillance décrites plus haut par référence aux figures 2 à 6.

La surveillance du programme ou de la partie de programme composé par les ensembles EI1, EI2 et EI3 procède alors comme suit.

20 L'unité de surveillance 22 se positionne d'abord en phase de début de surveillance (étape 30, figure 3).

Le déroulement débute par l'exécution du premier ensemble d'instructions EI1. La première instruction de surveillance placée en en-tête de cet ensemble sera d'abord chargée dans l'unité de surveillance 22. En réponse à cette instruction, l'unité
25 de surveillance initialise son compteur d'instructions et son registre 24 de valeur de hachage VH (étape 34, figure 3) et procède à la routine de hachage pour chaque instruction Inst.EI1-1 à Inst.EI1-j de premier ensemble d'instructions selon les étapes 36 à 46 de la figure 3.

La dernière instruction EI1-j de l'ensemble qui commande le branchement est
30 ainsi également soumise au hachage par l'unité de surveillance 22 avant d'être transmise vers le processeur 4.

L'instruction suivante est la seconde instruction de surveillance (étape 46, figure 3) en queue du premier ensemble d'instructions EI1, qui provoque la phase de

comparaison entre la dernière valeur de hachage enregistrée dans le registre 24 avec la valeur de référence $V_{réf}$ associée à cette seconde instruction.

S'il est détecté à l'étape 50 de comparaison (figure 3) que la dernière valeur de hachage ainsi enregistrée ne correspond pas à la valeur de référence $V_{réf}$, l'unité de surveillance 22 procède aux étapes d'interruption du programme 54 et 56 (figures 3 ou 4) ou 54 à 70 (figure 6). De préférence, on prévoit que cette interruption se produit avant même que le programme ait exécuté le branchement conditionnel. Ceci peut être réalisé, par exemple, en associant à l'instruction de saut avec une instruction d'attente de validation venant de l'unité de surveillance, en employant des techniques de programmation connues.

S'il est détecté à l'étape 50 (figure 2) de comparaison que la dernière valeur de hachage ainsi enregistrée correspond effectivement à la valeur de référence $V_{réf}$, l'unité de surveillance autorise l'exécution du branchement conditionnel déterminé par la dernière instruction EI1-j de l'ensemble. Le programme se poursuit alors vers l'un ou l'autre des deuxième ou troisième ensembles d'instructions conformément aux conditions de branchement posées par cette dernière instruction.

On suppose dans l'exemple que le saut conditionnel provoque le branchement vers le troisième ensemble d'instructions à exécuter. Dans ce cas, le compteur d'instructions 6 fait passer le pointeur d'instructions 8 directement de la seconde instruction de surveillance en queue du premier ensemble d'instructions EI1 à la première instruction de surveillance en tête du troisième ensemble d'instructions EI3.

L'unité de surveillance exécutera cette nouvelle première instruction en ré-initialisant le compteur d'instructions et le registre 24. La procédure de surveillance pour ce troisième ensemble d'instructions se poursuit donc exactement de la même manière que pour le premier ensemble d'instructions. Ainsi, l'unité de surveillance 22 procédera au hachage successif de chacune des instructions lues de cet ensemble, en commençant le hachage avec la même valeur "graine" (qui correspond ici à zéro) que pour le premier ensemble. La seconde instruction de surveillance permet cette fois de déceler un déroulement non-prévu dans l'exécution localisée au niveau de ce troisième ensemble d'instructions et de procéder au même type d'action à l'étape 56.

On comprendra que l'explication donnée pour le cas d'un branchement vers le troisième ensemble s'applique de manière rigoureusement analogue dans le cas d'un

branchement vers le deuxième ensemble d'instructions suite à l'exécution de l'instruction de branchement du premier ensemble d'instructions.

Il est possible de permettre à l'unité de surveillance 22 de comptabiliser non seulement le nombre de déroulements non-prévus dans un programme comportant
5 des sauts, mais également les ensembles d'instructions indépendamment surveillés dans lesquels ils se sont produits.

Ainsi, l'unité de surveillance 22 selon le deuxième mode de réalisation (figure 5) peut enregistrer dans sa mémoire 60 l'ensemble d'instructions concerné par chaque interruption constatée. Il est aussi possible d'établir des critères d'invalidation
10 d'usage futur du programme en fonction des ensembles d'instructions dans lesquels se sont les interruptions.

Bien entendu, on comprendra que l'unité de surveillance 22 peut être soit matérialisée séparément du processeur 4 ou intégrée fonctionnellement à ce dernier.

Enfin, il est clair que tous les aspects de l'invention décrits en termes de
15 procédé peuvent être aisément compris en termes de moyens matériels de leur mise en œuvre, et inversement. De même, on comprendra que l'invention décrite couvre également toutes les transpositions évidentes d'un mode de réalisation ou sa variante à un autre.

REVENDEICATIONS

1. Procédé de surveillance du déroulement de l'exécution d'une suite d'instructions (Inst.1-Inst.n) d'un programme informatique, consistant à analyser la séquence des instructions transmises vers le processeur (4) destiné à exécuter le programme surveillé et à vérifier le résultat de cette analyse par référence à des données de référence (Vréf) enregistrées avec ledit programme.

2. Procédé selon la revendication 1, caractérisé en ce que les données de référence comprennent une valeur (Vréf) préétablie de manière à correspondre au résultat de l'analyse réalisée lors du procédé de surveillance seulement si toutes les instructions (Inst.1-Inst.n) de la séquence d'instructions ont été effectivement analysées lors du déroulement du programme.

3. Procédé selon la revendication 1 ou 2, caractérisé en ce que ladite analyse de la séquence d'instructions (Inst.1-Inst.n) comprend l'extraction (38) d'une donnée de chaque instruction transmise vers le processeur (4) et un calcul prédéterminé (40, 42) sur chaque donnée ainsi extraite, et en ce que la vérification comprend une comparaison (50) du résultat de l'analyse avec les données de référence (Vréf).

4. Procédé selon l'une quelconque des revendications 1 à 3, caractérisé en ce que ladite vérification du résultat de l'analyse est provoquée par une instruction (Inst.n+1) placée à un emplacement prédéterminé dans le programme à surveiller, cette instruction contenant les données de référence (Vréf) relatives à un ensemble d'instructions (Inst.1-Inst.n) dont l'exécution correcte est à surveiller.

5. Procédé selon l'une quelconque des revendications 1 à 4, caractérisé en ce que, lorsque les instructions (Inst.1-Inst.n) de l'ensemble d'instructions à surveiller se présentent sous la forme d'une valeur, par exemple des codes enregistrés sous forme hexadécimal ou décimal, on effectue ladite analyse des instructions en considérant celles-ci en tant que valeur numérique.

6. Procédé selon la revendication 1, comprenant les étapes consistant à :

- lors de la préparation du programme à surveiller :

5 - incorporer, à au moins un emplacement prédéterminé d'une séquence d'instructions (Inst.1-Inst.n) du programme, une valeur de référence (Vréf) établie selon une règle prédéterminée appliquée sur des données identifiables dans chaque instruction à surveiller, et

- lors de l'exécution du programme à surveiller:

10 - obtenir (38) lesdites données identifiables dans chaque instruction reçue en vu de son exécution,

- appliquer (40, 42) ladite règle prédéterminée sur lesdites données identifiables ainsi obtenues pour établir une valeur de vérification (VHn), et

- vérifier (50) que cette valeur de vérification correspond effectivement à la valeur de référence enregistrée avec le programme.

15

7. Procédé selon l'une quelconque des revendications 1 à 6, caractérisé en ce qu'il comprend en outre une étape (56) d'interruption du déroulement du programme surveillé si l'analyse révèle le programme surveillé ne s'est pas déroulé de manière prévue.

20

8. Procédé selon l'une quelconque des revendications 1 à 7, caractérisé en ce qu'il comprend en outre une étape (70) d'invalidation pour usage futur du dispositif comprenant le programme surveillé si ladite analyse révèle un nombre prédéterminé de fois que le programme surveillé ne s'est pas déroulé de manière prévue.

25

9. Procédé selon l'une quelconque des revendications 1 à 8, caractérisé en ce l'ensemble d'instructions à surveiller ne comporte pas de sauts dans son déroulement prévu.

30

10. Procédé selon l'une quelconque des revendications 1 à 8, caractérisé en ce que, lorsque le programme (EI1, EI2, EI3) ou la portion de programme à surveiller prévoit au moins un saut, on applique le procédé de surveillance séparément sur des

ensembles d'instructions de ce programme qui ne comportent pas de sauts entre deux instructions successives.

11. Procédé selon la revendication 10, caractérisé en ce que lorsque le
5 programme à surveiller comporte une instruction (EI1-j) donnant lieu à un saut dépendant des données manipulées, on met en œuvre le procédé de surveillance séparément pour un ensemble d'instructions (EI1) qui précède le saut, et pour au moins un ensemble d'instructions (EI2, EI3) qui succède à ce saut.

10 12. Procédé selon la revendication 11, caractérisé en ce que, pour un ensemble d'instructions (EI1) prévoyant un saut, on intègre à cet ensemble l'instruction (EI1-j) qui commande ce saut aux fins de l'analyse visant à obtenir la valeur de vérification (VH) de cet ensemble d'instructions, et on vérifie ainsi le bon déroulement de cet ensemble d'instructions avant d'exécuter l'instruction de saut.

15

13. Procédé selon l'une quelconque des revendications 1 à 12, caractérisé en ce que l'on ré-initialise l'analyse avant chaque nouvelle surveillance d'une séquence ou d'un ensemble (EI1, EI2, EI3) d'instructions à surveiller.

20

14. Procédé selon la revendication 13, caractérisé en ce que la ré-initialisation de l'analyse à chaque nouvelle surveillance consiste à effacer ou remplacer une valeur de vérification (VH) obtenue lors d'une précédente analyse.

15. Procédé selon la revendication 13 ou 14, caractérisé en ce que la ré-
25 initialisation de l'analyse de surveillance est commandée par le logiciel protégé lui-même.

16. Procédé selon l'une quelconque des revendications 1 à 15, caractérisé en ce que l'analyse produit une valeur de vérification (VH) obtenue en tant que dernière
30 valeur d'une suite de valeurs que l'on fait évoluer successivement avec l'analyse de chacune des instructions (Inst.1-Inst.n) analysées de l'ensemble d'instructions, permettant ainsi de contenir un état interne du déroulement du procédé de surveillance et de suivre son évolution.

17. Procédé selon l'une quelconque des revendications 1 à 16, caractérisé en ce que l'analyse consiste à calculer (40, 42), pour chaque instruction considérée (Inst.n) succédant à une instruction précédente (Inst.n-1), le résultat d'une opération sur à la fois une valeur (VHn) obtenue de l'instruction considérée et le résultat (VHn-1) obtenu par la même opération effectuée sur l'instruction précédente.

18. Procédé selon l'une quelconque des revendications 1 à 17, caractérisé en ce que l'analyse consiste à appliquer de manière réursive une fonction de hachage $f(VHn-1, Vinst.n)$ sur des valeurs obtenues de chaque instruction surveillée, en partant d'une dernière initialisation effectuée.

19. Procédé selon l'une quelconque des revendications 1 à 17, caractérisé en ce que l'analyse consiste à faire évoluer une valeur de vérification en effectuant un calcul de redondance non nécessairement cryptographique sur l'ensemble des codes d'opération et des adresses exécutées depuis la dernière initialisation effectuée.

20. Procédé selon l'une quelconque des revendications 1 à 19, caractérisé en ce que l'analyse consiste à obtenir une valeur de comparaison (VCn) par calcul de valeurs intermédiaires successives au fur et à mesure que l'on obtient les données des instructions respectives servant pour ce calcul durant l'exécution de ces instructions.

21. Procédé selon l'une quelconque des revendications 1 à 19, caractérisé en ce que l'analyse comprend une étape de sauvegarde de chaque donnée nécessaire pour la vérification, obtenue à partir des instructions de l'ensemble d'instructions à surveiller (Inst.1-Inst.n) au fur et à mesure qu'elles sont exécutées, et de n'effectuer un calcul de la valeur de vérification (VHn) à partir de ces données seulement au moment nécessaire, une fois que toutes les données nécessaires ont été obtenues.

22. Dispositif de surveillance du déroulement de l'exécution d'une suite d'instructions (Inst.1-Inst.n) d'un programme informatique, comportant des moyens (22-26) pour analyser la séquence des instructions transmises vers le processeur (4) destiné à exécuter le programme surveillé et des moyens (26) pour vérifier le résultat

(VCn) de cette analyse par référence à des données de référence (Vréf) enregistrées avec ledit programme.

23. Dispositif selon la revendication 22, adapté pour mettre en œuvre le
5 procédé de surveillance selon l'une quelconque des revendications 1 à 21, caractérisé en ce qu'il comporte un registre (24) permettant d'enregistrer des résultats intermédiaires (VH) dans un calcul en chaîne effectué par le moyen d'analyse (26) pour obtenir une valeur de vérification (VHn).

10 24. Dispositif selon la revendication 23, caractérisé en ce qu'il comprend des moyens pour permettre l'enregistrement d'une valeur prédéterminée ou une remise à zéro du registre (24) sous la commande d'une instruction (Inst.n+1) transmise lors de l'exécution d'un programme à surveiller, par exemple à l'occasion d'un saut dans le programme.

15 25. Dispositif selon l'une quelconque des revendications 22 à 24, caractérisé en ce qu'il comporte un moyen (60) de comptabilisation du nombre de déroulements non-prévus du programme surveillé, tel que déterminé par le moyen d'analyse (26), et des moyens pour invalider l'utilisation future du programme à surveiller si ce
20 nombre atteint un seuil (VCseuil) prédéterminé.

26. Dispositif selon l'une quelconque des revendications 22 à 25, caractérisé en ce qu'il est intégré à un dispositif programmé, telle qu'une carte à puce, contenant ledit programme à surveiller.

25 27. Dispositif selon l'une quelconque des revendications 22 à 25, caractérisé en ce qu'il est intégré un dispositif d'exécution de programme (20).

28. Dispositif d'exécution de programme (20), destiné à exécuter une suite
30 d'instructions (Inst.1-Inst.n) d'un programme informatique, caractérisé en ce qu'il comporte des moyens (22-26) pour analyser la séquence des instructions transmises pour exécution et des moyens pour vérifier le résultat de cette analyse par référence à des données de référence (Vréf) enregistrées avec le programme à surveiller.

29. Dispositif d'exécution de programme (20) selon la revendication 28, adapté pour mettre en œuvre le procédé selon l'une quelconque des revendications 1 à 21.

5

30. Dispositif programmé comportant une suite d'instructions enregistrées (Inst.1-Inst.n), caractérisé en ce qu'il comporte en outre des données de référence (Vréf) préétablies en fonction de données contenues dans lesdites instructions et destinées à permettre une vérification de la séquence des instructions analysées selon l'une quelconque des revendications 1 à 21.

10

31. Dispositif selon la revendication 30, caractérisé en ce qu'il se présente sous forme de carte à puce.

15

32. Dispositif selon la revendication 30 ou 31, caractérisé en ce que les données de référence (Vréf) sont enregistrées sous la forme de valeur(s) pré-câblée(s) fixée(s) en mémoire.

20

33. Dispositif de programmation d'un dispositif destiné à être programmé selon l'une quelconque des revendications 30 à 32, caractérisé en ce qu'il comprend des moyens pour inscrire à au moins un emplacement prédéterminé d'une séquence d'instructions du programme (Inst.1-Inst.n) une valeur de référence (Vréf) calculée selon un mode préétabli à partir de données comprises dans chaque instruction d'un ensemble d'instructions dont on souhaite surveiller l'exécution.

25

34. Machine virtuelle ou interpréteur interprétant un code critique, caractérisé en ce qu'il met en œuvre le procédé selon l'une quelconque des revendications 1 à 21 pour l'exécution de ce code critique.

29. Dispositif d'exécution de programme (20) selon la revendication 28, adapté pour mettre en œuvre le procédé selon l'une quelconque des revendications 1 à 21.

5

30. Dispositif programmé comportant une suite d'instructions enregistrées (Inst.1-Inst.n), caractérisé en ce qu'il comporte en outre une mémoire fixe contenant des données de référence (Vréf) préétablies en fonction de données contenues dans lesdites instructions et destinées à permettre une vérification de la séquence des instructions analysées selon l'une quelconque des revendications 1 à 21, ledit
10 dispositif étant destiné à coopérer avec un dispositif de surveillance selon l'une quelconque des revendications 22 à 29.

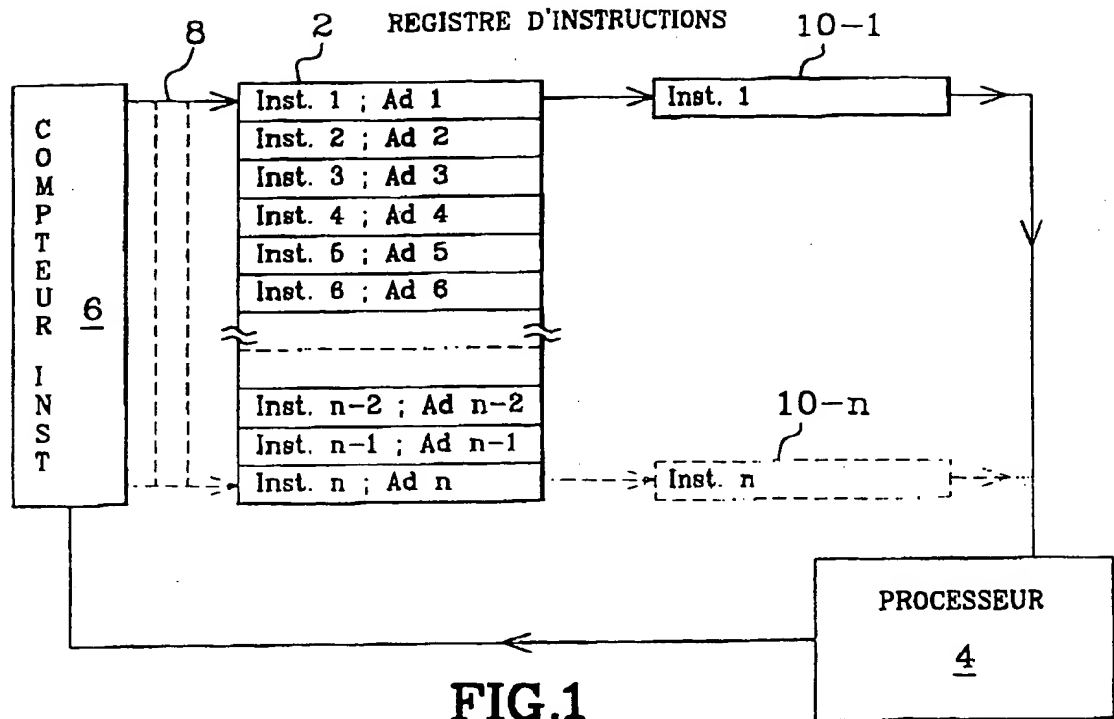
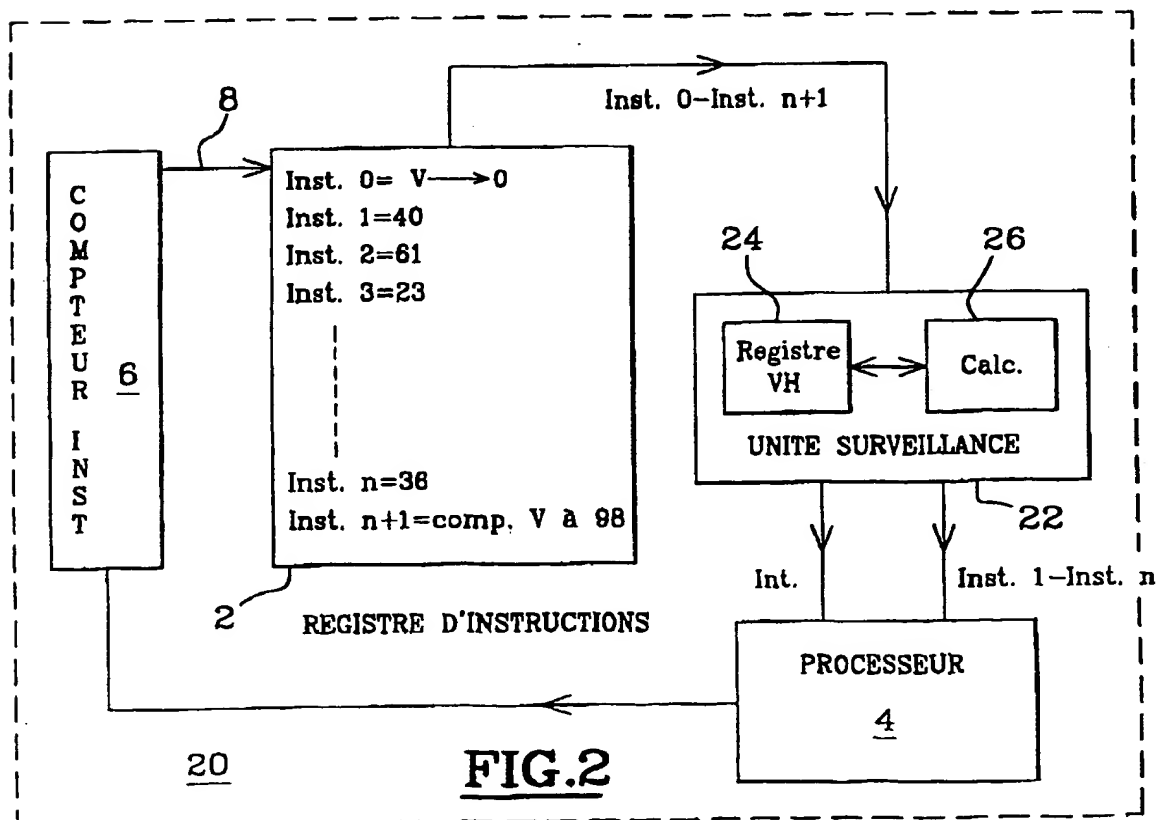
31. Dispositif selon la revendication 30, caractérisé en ce qu'il se présente
15 sous forme de carte à puce.

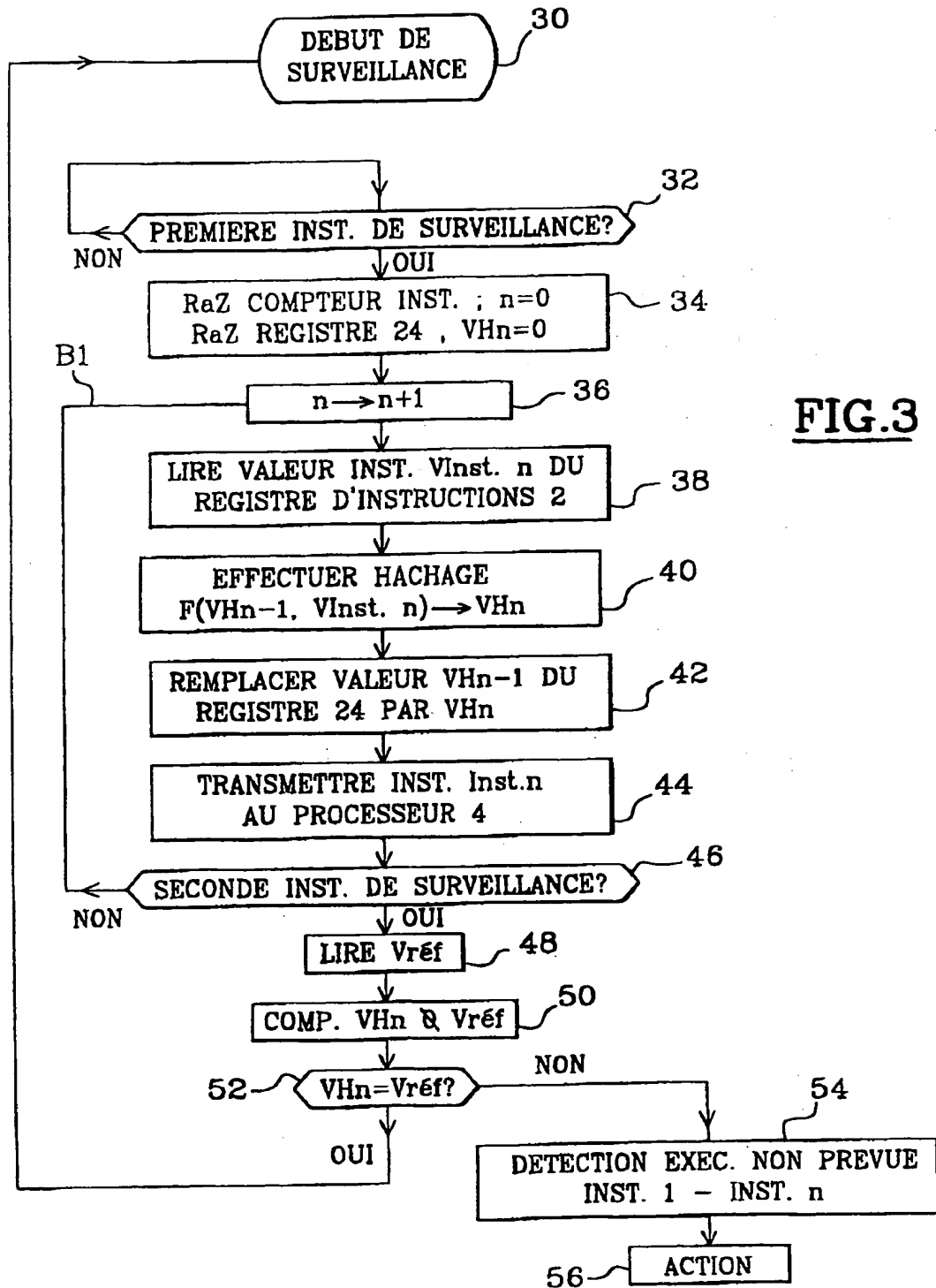
32. Dispositif selon la revendication 30 ou 31, caractérisé en ce que les données de référence (Vréf) sont enregistrées sous la forme de valeur(s) pré-câblée(s) fixée(s) en mémoire.

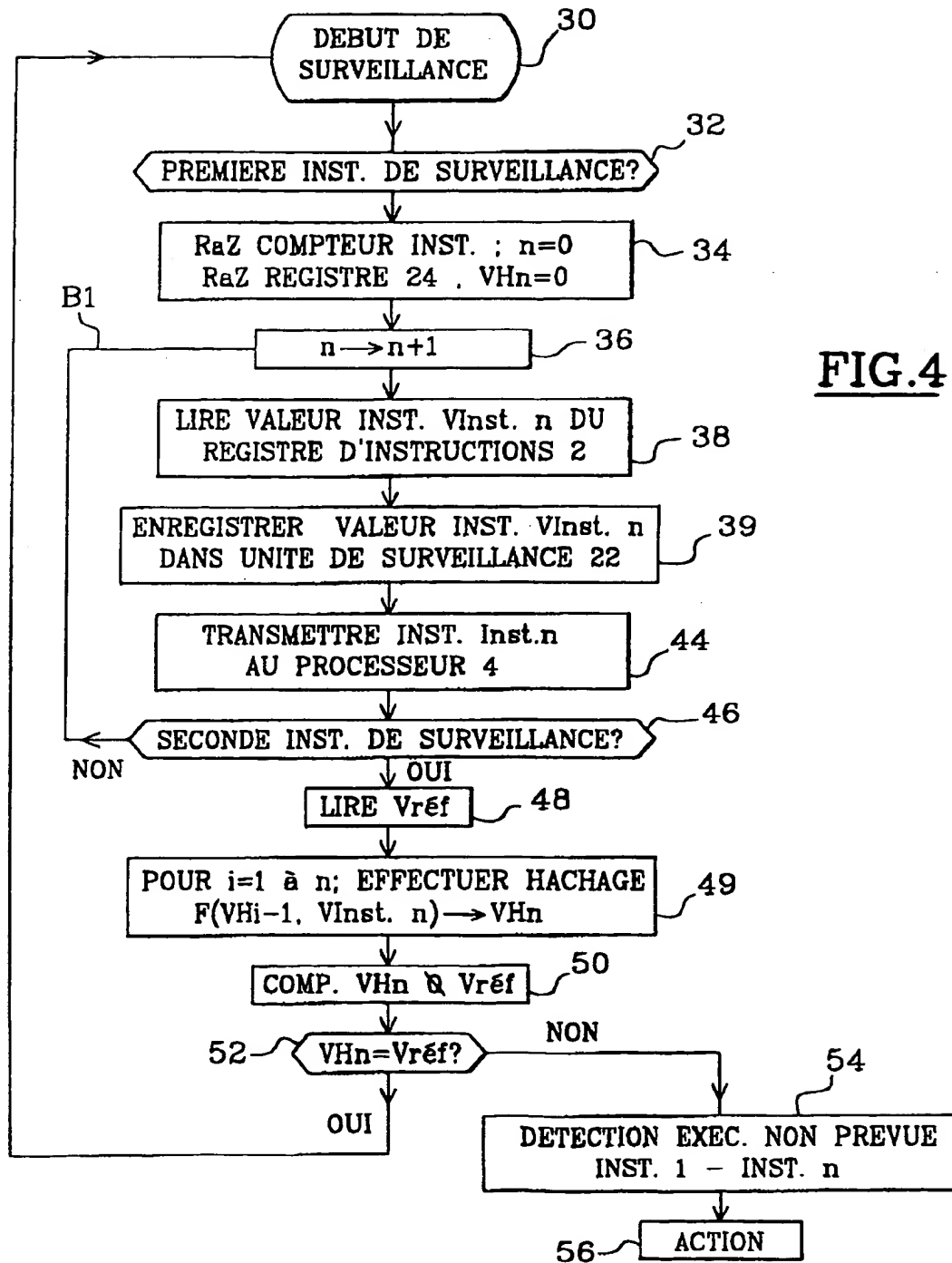
20

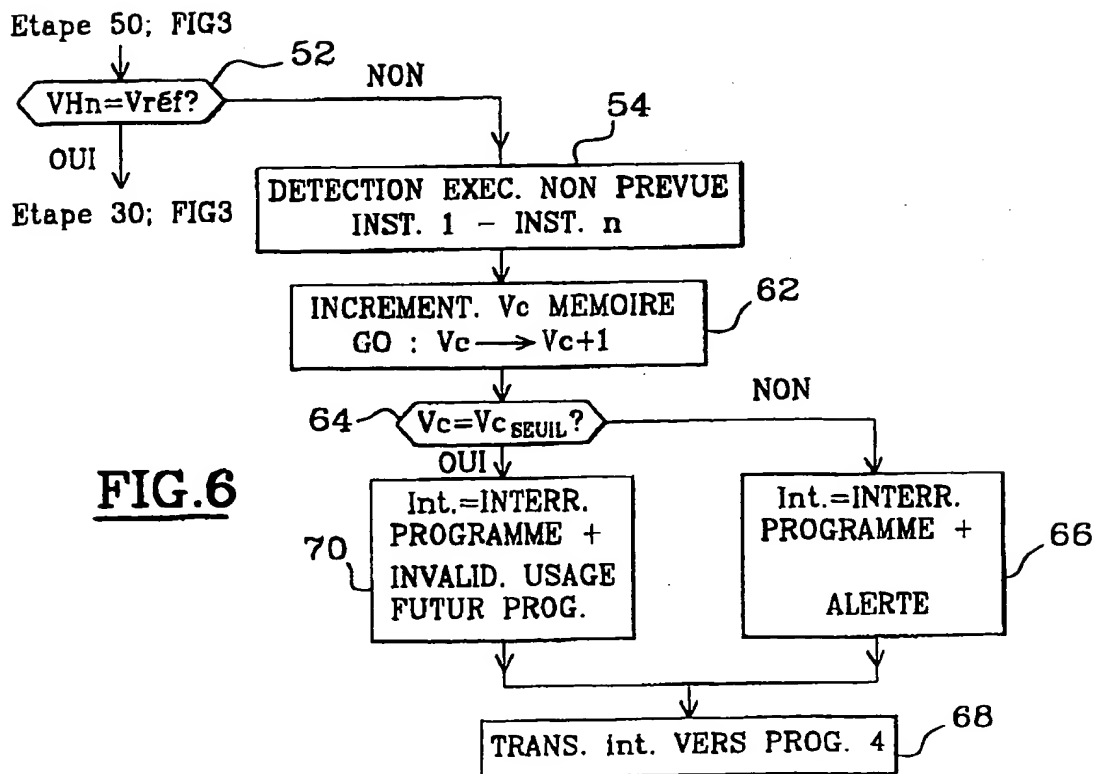
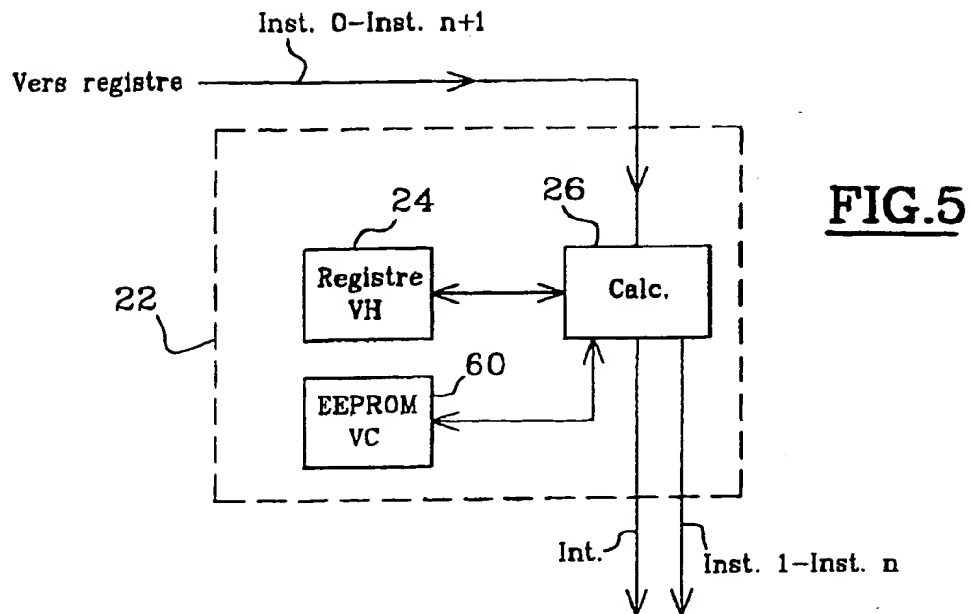
33. Dispositif de programmation d'un dispositif destiné à être programmé selon l'une quelconque des revendications 30 à 32, caractérisé en ce qu'il comprend des moyens pour inscrire à au moins un emplacement prédéterminé d'une séquence d'instructions du programme (Inst.1-Inst.n) une valeur de référence (Vréf) calculée
25 selon un mode préétabli à partir de données comprises dans chaque instruction d'un ensemble d'instructions dont on souhaite surveiller l'exécution.

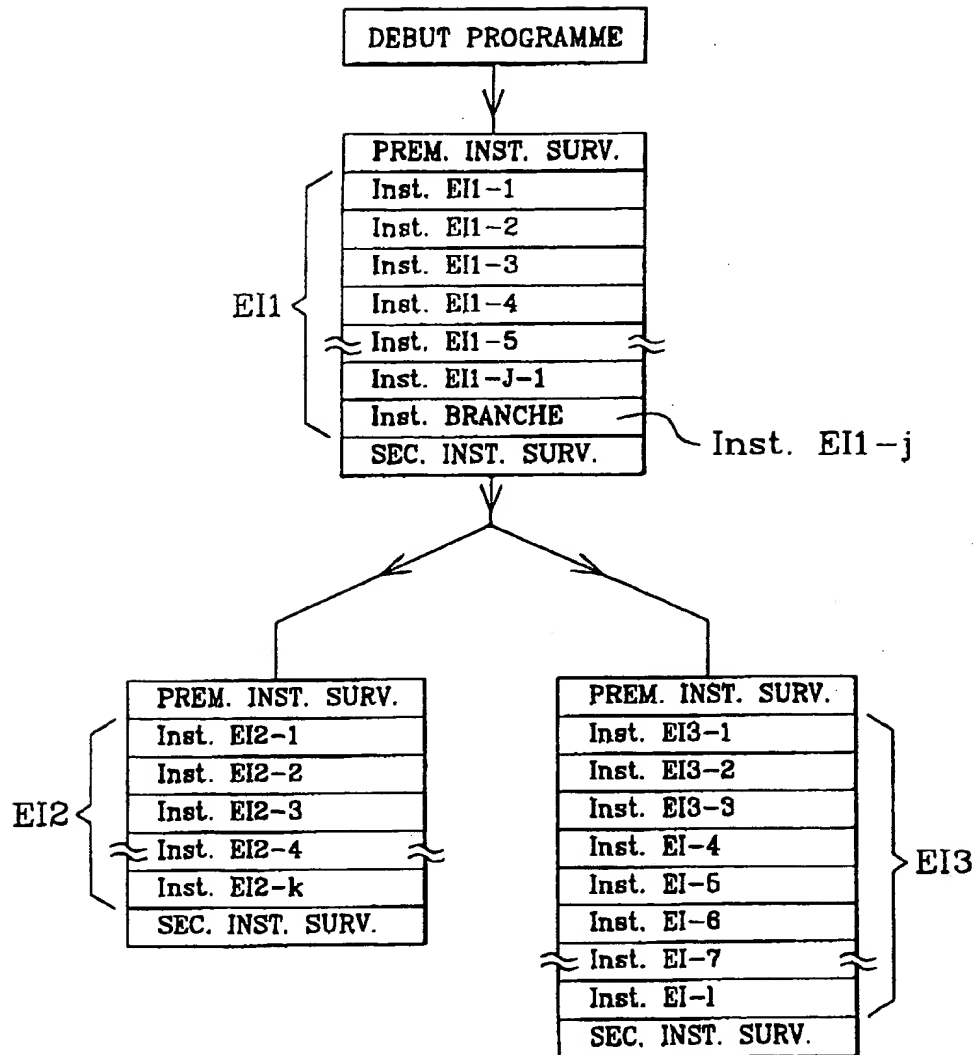
34. Machine virtuelle ou interpréteur interprétant un code critique, caractérisé en ce qu'il met en œuvre le procédé selon l'une quelconque des
30 revendications 1 à 21 pour l'exécution de ce code critique.

**FIG. 1****FIG. 2**







**FIG.7**